International Conference on Latest Innovation in Engineering Science and Management
Buddha Institute of Technology, GIDA, Gorakhpur (UP)
30th - 31st March 2019 www.conferenceworld.in ISBN : 978-93-87793-80-4

Concurrency Control in Distributed database management systems: Dealing Unfair Transactions at Higher Access Classes (DUTHAC)

Underguided-Mr.Alok Kumar Srivastava(Assistant Professor)
Bhavya Singh[1]*, Anjali Singh[2], Neha Singh[3], Priya Singh[4]
Department Of Information Technology
Buddha Institute of Technology
Gida,Gorakhpur UP India

Abstract-

A secure concurrency control used for maintaining consistency of the database and convert arise due to data conflicts between transactions. The existing secure concurrency control is a way of dealing unfair transactions at higher access classes .In this paper, a secure two phase locking protocol is presented, which is shown to be free to convert channels arises due to data conflicts between transactions which gives responsibility for execution of all transactions of their accessed class. Privacy and data security have been issues and this paper addresses first the protection of organizational data that has to be reassessed.
Keywords: Serialization, Two-phase locking, Timestamp, Multiversion Timestamp Ordering, Concurrency Control, Deadlock.

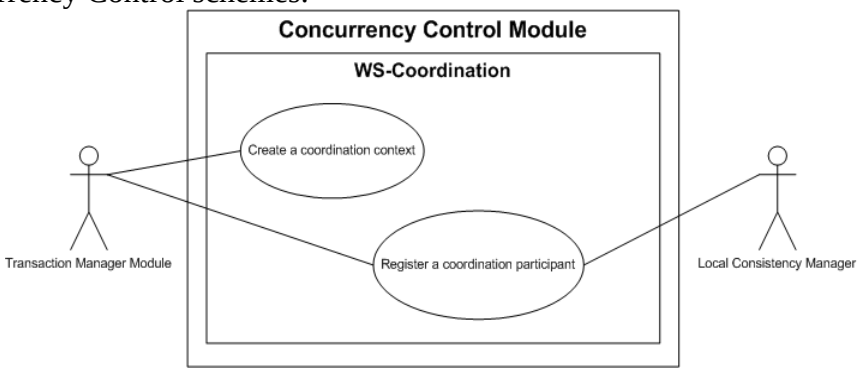
1. INTRODUCTION

It is one of the problems in Distributed database management systems (DDBMS) is Concurrency control in a transactions. The concurrency control techniques described into several ways: locking (Two-Phase Locking) and ordering timestamp. The two methods delays of transactions in the system is in effect when the system is saturated with conflicts .That's why the Two-phase locking (2PL) protocol is suitable to used . But this model has a main drawback is the possibility of deadlocks appearance and also Deadlock resolution and recovery from system failures. There are various applications that require to access database which is located in different databases. Local transactions are executed under the control of local database management systems (DBMSs) while global transactions access DBMSs are the control of Global Transaction Manager (GTM). A global transaction is to divide into sub-transactions, each one is considered as a local transaction by using local DBMS. Various protocols have been proposed by ACID properties.

2. LITERATURE REVIEW

When several transactions execute concurrently in database, the consistency of data may no longer be preserved. It is necessary for the system to control the interaction among the

concurrent transactions, and this control is achieved through one variety of mechanisms called Concurrency Control schemes.



Correctness Criteria for Secure Schedulers, Convert channel analysis and removal is the single most important issue in multilevel secure concurrency control .The notation of non interference has been proposed[2]. Secure concurrency control mechanism and the necessary and sufficient conditions for a secure, interference-free scheduler. Three of these properties are of relevance to the secure two phase locking protocol these are:

1. Value Security: A scheduler satisfies this property if values read by a subject are not affected by actions with higher subject classification levels.
2. Delay Security: This property ensures that the delay experienced by an action is not affected by the actions of a subject at a higher classification level.
3. Recovery Security: A set of transactions is in a deadlock state when every transaction in the set is waiting for an event that can only be caused by another transaction in the set [3].

Approaches to Secure Concurrency Control

1. Locking

A locking protocol is a set of rules that state when transaction may lock and unlock each of data items in database.

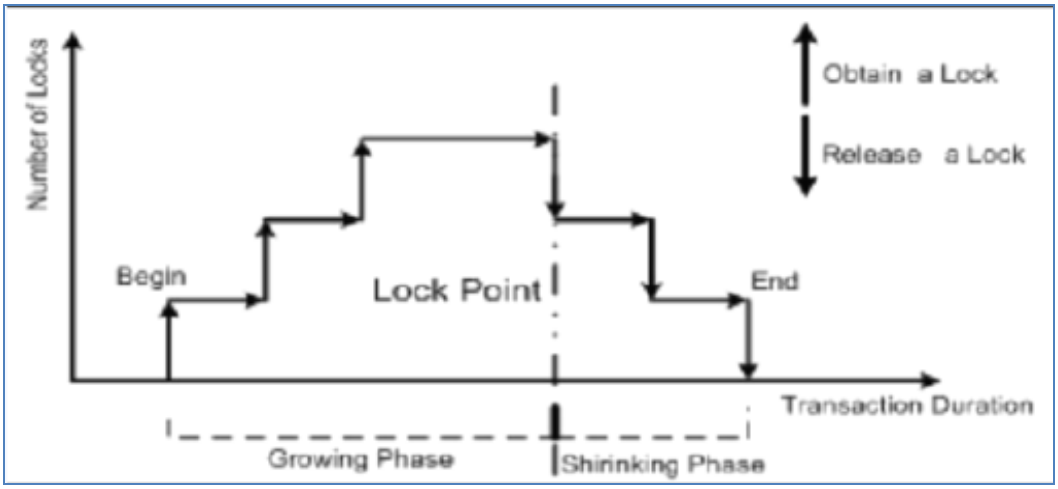
1.1 Phase Locking

In databases and transaction processing two-phase locking, (2PL) is a concurrency control that guarantees serializability. It is also called in histories the resulting set of database transaction schedules. The protocol utilizes locks that block other transactions from accessing the same data during a transaction's life.
Two phase locking protocol ensures serializability this protocol required that each transaction issue lock and unlock requests.

Two phase locking protocol are as follows:-

1. Growing phase: A transaction may obtain locks, but may not release any lock. It is also called Expanding phase.
2. Shrinking phase: A transaction may release locks, but may not obtain any new locks. It is

also called contracting phase.



There are mainly two modes on which data item may lock:-

1. Shared lock: It is denoted by S, if transaction Ti has obtain a shared lock on data item Q, then Ti can read but can't write Q .It is also called Read-lock.

2. Exclusive lock: It is denoted by X, if a transaction Tj has mode lock on data item Q then Tj can both read and write Q .It is also called Write–lock.

The basic rules for locking are:-

- If transaction has a read-lock on a data item, it can read the data item but can't update the data item.
- If transaction has a read-lock on a data item, other transaction can obtain a read-lock on data item but not write lock.
- If transaction has write-lock, it can both read and update the data item.
- If transaction has write -lock on data item then other transaction can't lock on that data item ,obtain either a read-lock or write-lock on that data item.

Lock compatibility table

Lock type	*read-lock*	*write-lock*
Read-lock		X
Write-lock	X	X

where , X indicates incompatibility.

1.2 Conservative two-phase locking

The difference between 2PL and C2PL is that C2PL's transactions obtain all the locks they need before the transactions begins .This is to ensure that a transaction that already holds

ISBN : 978-93-87793-80-4

some locks will not block waiting for other locks. Conservative 2PL prevents deadlocks.

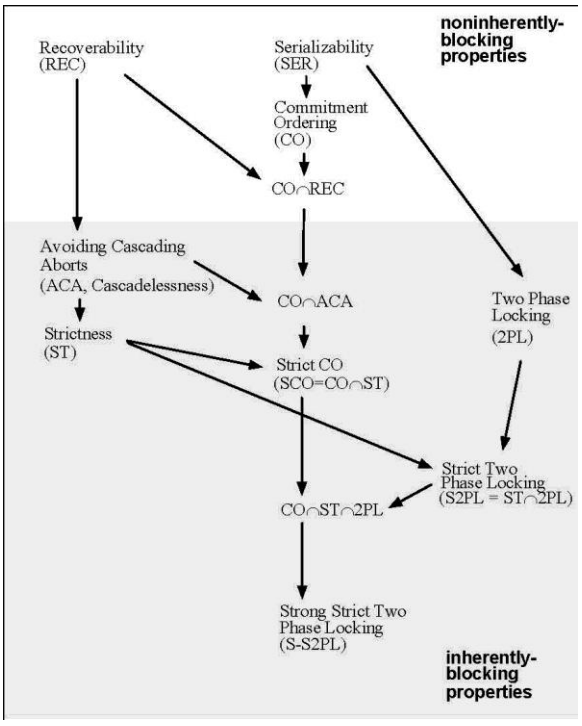
1.3 Strict Two-Phase Locking

Strict Two-Phase locking protocol is modification of Two- Phase locking protocol in which Cascading rollbacks can be avoided . The strict two-phase locking protocol requires that in addition to locking being two-phase, all exclusive-mode locks taken by a transaction must be held until that transaction commits.

It is proven to be an effective mechanism in many situations, and provides besides Serializability also Strictness. To comply with the S2PL protocol a transaction needs to comply with 2PL, and release its write (exclusive) locks only after it has ended, i.e., being either committed or aborted. On the other hand, read (shared) locks are released regularly during phase 2.Implementing general S2PL requires explicit support of phase-1 end, separate from transaction end, and no such widely utilized product implementation is known.

Strong strict two-phase locking is a special case of strict two-phase locking class of schedules is a proper subclass of strict two-phase locking.

Locking will fail in a secure database because the security properties prevent actions in a transaction t_i at a higher Access class form delaying actions in a transaction T2 at a lower access class.



2. Timestamp

every conflicting read and write operation are executed in timestamp order. Use the value of system clock as timestamping and also use logical counter that is incremented after a new timestamp has been assigned.

ISBN : 978-93-87793-80-4

Implement scheme , we associate timestamping values to each data item Q:

1.*W-timestamp(Q)* Any transaction that executed write(Q)successfully and denotes the largest timestamp .
2.*R-timestamp(Q)* Any transaction that executed read(Q)successfully and denotes the largest timestamp .

The timestamp-ordering protocol

1. Suppose that transaction Ti issues read (Q)
   - If TS(T_i)< W-timestamp(Q),then T_i need to read a value of Q that was already overwritten. Hence, read operation is rejected and T_i is rollback.
   - If TS(T_i)>= W-timestamp(Q),then the read operation is executed, and R-timestamp(Q) is set to maximum of R-timestamp(Q) and TS(T_i).

2. Suppose that transaction T_i issues write(Q)
   - If TS(T_i)< R-timestamp(Q) ,then the value of Q that T_i is producing was needed previously ,and the system assumed that value would never produced. Hence, the system reject the write operation and roll T_i back.
   - If TS(T_i)< W-timestamp(Q) ,then T_i is attempting to write an obsolete value Q .Hence, the system reject the write operation and roll T_i back.
   - Otherwise, the system execute the write operation and sets W-timestamp(Q) to TS (T_i)

Timestamp-ordering fails for similar reasons as locking, with timestamps taking the role of locks ,since a transaction at a higher access class cannot cause the abortion of any other transaction at a lower access class.[1]

3. Optimistic Concurrency Control

Whenever a conflict is detected between a transaction that a higher access class in its validation phase and a transaction T_i at a lower access class, the transaction at the higher access class is aborted, while the transaction at the lower access class is not affected.
A major problem with using optimistic concurrency control is the possible starvation of higher-level transactions.

4. Multiversion Timestamp Ordering

A secure version of the **MVTO** scheduler is presented in [4]. The difference between secure MVTO and basic MVTO is that secure MVTO will sometimes assign a new transaction a

timestamp that is earlier than the current timestamp. Active transactions moves effectively with respect to past transaction. When a transaction **begins ,** it is assigned a timestamp that precedes the timestamps of all transactions active at strictly dominated access classes and that follows the timestamp of all transactions at its own access class. This approach to timestamp assignment is what makes it impossible for a transaction to invalidate a read from a higher access class .This method has the drawback that transactions at a higher access class are forced to read arbitrarily old values from the database due to the timestamp assignment. This problem can be especially serious if most of the lower level transactions are long running transactions.

Deadlock detection and Recovery

Deadlock detection

Deadlock can't be avoided in Two-Phase locking. For **example** . consider the input schedule below:

T_i (SECRET) : r[x] r[y]
T2 (UNCLASSIFIED) : w[y] w[x]

The sequence of operations performed are rl1[x] r1[x] wl2[y] w2[y] vwl2[x] c2. After these operations, deadlock would occur because r1[y] waits for w2[w2y] to release its lock and vw2[x] waits for r1[x] to release its lock. Deadlocks can be detected by constructing a waits-for graph [Bern 1]. A scheduler recovers from a deadlock by aborting one of the transactions in the cycle. A secure scheduler must ensure that recovery security is not violated by the choice of a victim. This can be guaranteed by aborting a transaction at the highest access class in the cycle detected in the waits-for graph.

Recovery of Deadlock

When a detection algorithm determines that a deadlock exist ,the system must recover from a deadlock . Atomicity property must satisfy by transaction in database system .All the effects must be present in the database by 'commit' transaction and all it's effect must be removed from the database, transaction is abort. Restart operation must be support by the system, which may be sent by the operating system upon recovery from system failure. The task restart maintains the consistent state by removing the effect of uncommitted transactions and apply missing effect on committed one. Restart also has to ensure that the order in which the missing effects are applied is the same as the order in which they were scheduled before the failure.
The properties mentioned above are not satisfied by the secure two phase locking protocol because a transaction may commit before all its effects are present in the database.

Consider the example shown:

T1(SECRET) : r[x] C1
T2(UNCLASSIFIED) : w[x] C2

When W2[x] arrives, it is forced to take a virtual lock on x and is put on the queue of actions

waiting to lock x. The scheduler cannot delay the commit of T2 because doing so would be a violation of delay security. Hence, T2 commits before w 2[x] executed on the actual data item x in stable storage. If a system failure were to occur immediately after T2 commits, the value to be written by W2 [x] could be lost forever because the lock tables are destroyed when the system fails.

The problem can be overcome by a modification to the UndoRedo algorithm [Bern 1]. Assume that the recovery manager (RM) stores a physical log of all writes in the form [T_i, x, v, before-image] where T_i is the transaction which wrote value v into data item .x. The before-image is the value of x before the write. Assume that an additional log (called the virtual-writes log) with entries of the form (T_i, x, vx, y, vy,...] is maintained by the RM. Each such entry consists of a transaction-id T_i, of a transaction that has committed, a list of data items on which T_i's writes are yet to be performed (even though T, has committed) as well as the values to be written by T_i to these data items.

Good Concurrency Protocol

An ideal concurrency control DBMS mechanism has the following objectives:

- Must be resilient to site and communication failures.
- It allows the parallel execution of transactions to achieve maximum concurrency.
- Its storage mechanisms and computational methods should be modest to minimize overhead.
- It must enforce some constraints on the structure of atomic actions of transactions.

3. CHALLENGES AND CURRENT STATUS OF CONCURRENCY CONTROL IN DISTRIBUTED DATABASE MANAGEMENT SYSTEMS

Very much challenges are present when we talking about concurrency control for any transaction that uses distributed database. The main issue is of deadlock. The transactions are waiting for long time for acquiring the resources. The main aim of this paper is to remove the concurrency problem for the resources that uses the distributed database.

When you introduce a locking protocol, deadlocks always become a possibility. T1 T2 WL(A) WL(B) R(A) R(B) WL(B) not granted WL(A) not granted. There are many ways to recognize deadlock – precedence graphs. The most common is to put a timeout on the transaction – if the locks haven't been granted in X amount of time, roll the transaction back to the beginning.

ISBN : 978-93-87793-80-4

T1	T2
WL(A)	
	WL(B)
R(A)	
	R(B)
WL(B) not Granted	
	WL(A) not granted

4. FUTURE GROWTH OF CONCURRENCY CONTROL IN DISTRIBUTED DATABASE MANAGEMENT SYSTEMS

Future work involves an implementation of the secure 2PL protocol and a comparative study of the characteristics of the secure 2PL. We can also started a work on the interaction between the security and realtime requirements of database system.
Very much challenges are present when we talking about concurrency control for any transaction that uses distributed database. The main issue is of deadlock. The transactions are waiting for long time for acquiring the resources. The main aim of this paper is to remove the concurrency problem for the resources that uses the distributed database.

5. Conclusion

In this paper a complete specification of a secure version of the two phase locking protocol with secure deadlock correction was given. The interaction between the protocol and the standard failure correction procedures will be discussed and modified to the commit and restart procedures were proposed.
Future work involves an implementation of the secure 2PL protocol and a comparative study of the characteristics of the secure 2PL. We can also started a work on the interaction between the security and realtime requirements of database system.

REFERENCES

[1] P. A. Bemstein, **N.** Goodman & V. Hadzilacos. “Concurrency Control and Recovery in Database Systems”, Reading, MA: Addison Wesley,1987.
[2] J. A. Goguen and J. Meseguer. “Security Policy and Security Models”, Proceedings of the IEEE Symposium on Security and Privacy, pp11-20, 1982.
[3] T. E Keefe, W. T. Tsai & J. Srivastava. “Multilevel Secure Database Concurrency Control”, In Proceedings of the Sixth International on Data Conference Engineering, pp 337-344, Los Angeles, CA, February 1990.
[4] T. F. Keefe & W. T. Tsai. “Multiversion Concurrency Control for multilevel Secure Database Systems”, Proceedings of the 11th IEEE Symposium on Security and Privacy, Oakland,California, pp 369-383, April 1990.

ISBN : 978-93-87793-80-4

[5] Rasikan David and Sang H. Son , Secure Two Phase Locking Protocol ,University of Virginia, Charlottesville, VA 22903,05 June 2014.

[6] S. Greenberg and D. Marwood. Real time groupware as a distributed system: concurrency control and its effect on the interface. In Proc. ACM Conference on Computer Supported Cooperative Work, pages 207– 217. ACM Press, Nov. 1994S

[7] J. Gray, A. Reuter. Transaction processing:Concepts and Techniques, Morgan Kaufmann Publishers, USA, 1993.